Spring 2012

Improving Search Results with Automated Summarization and Sentence Clustering

Steven Cotter

IMPROVING SEARCH RESULTS WITH AUTOMATED SUMMARIZATION AND SENTENCE CLUSTERING

A Thesis

Submitted to the Department of Mathematics and Computer Science

McAnulty College and Graduate School of Liberal Arts

Duquesne University

In partial fulfillment of the requirements for

the degree of Master of Science

By

Steven Cotter

March 2012

IMPROVING SEARCH RESULTS WITH AUTOMATED SUMMARIZATION AND SENTENCE CLUSTERING

By

Steven Cotter

March 22, 2012

APPROVED_____
Patrick Juola Ph.D., Associate Professor
Department of Mathematics & Computer Science

APPROVED_____
John Kern, Ph.D., Associate Professor
Department of Mathematics & Computer Science

APPROVED_____
Donald Simon, Ph.D., Graduate Director, Computational Mathematics
Department of Mathematics & Computer Science

APPROVED_____
James Swindal, Ph.D., Acting Dean
McAnulty College and Graduate School of Liberal Arts

ABSTRACT

IMPROVING SEARCH RESULTS WITH AUTOMATED SUMMARIZATION AND SENTENCE CLUSTERING

By

Steven Cotter

March 2012

Thesis supervised by Patrick Juola.

Have you ever searched for something on the web and been overloaded with irrelevant results? Many search engines tend to cast a very wide net and rely on ranking to show you the relevant results first. But, this doesn't always work. Perhaps the occurrence of irrelevant results could be reduced if we could eliminate the unimportant content from each webpage while indexing. Instead of casting a wide net, maybe we can make the net smarter. Here, I investigate the feasibility of using automated document summarization and clustering to do just that. The results indicate that such methods can make search engines more precise, more efficient, and faster, but not without costs.

ACKNOWLEDGEMENT

I would like to thank Dr. Patrick Juola for his support and guidance as my thesis supervisor. Additionally, I am grateful to Dr. John Kern and Dr. Donald Simon for their comments and guidance, to Dr. James Swindal for providing funds to purchase test data, and to Matt Oberlander for making everything work. Special thanks also go to my supervisor, Eric Myers, for providing the flexibility for me to pursue this degree full-time while working full-time and for providing invaluable advice for this thesis, my career, and my life.

I would like to thank my family and loved ones, especially Lindsay, for their support and understanding over the last two years, which has allowed me to complete this program.

I am greatly indebted to all the faculty and students of the Mathematics and Computer Science Department at Duquesne University. Their passion for the power and possibility offered by computation and technology in each of their respective fields is inspiring.

Before I began this program, I would arrive to work feeling that I was the apparatus at my job, not my computer. I was a slave to it. Now, I have a feeling of omnipotence. Not only can I make this machine do anything I want, but I can add to it. I can make it better. This feeling is more valuable than any degree.

TABLE OF CONTENTS

Page

LIST OF TABLES

Page

LIST OF FIGURES

Page

Chapter 1

Introduction

Have you ever searched for something on the web and been overloaded with irrelevant results? Many search engines tend to cast a very wide net and rely on ranking to show you the relevant results first. But often times these ranking algorithms leave much to be desired. Relevant results aren't useful if they are distributed over a massive number of irrelevant results. Thus, this tendency toward casting a very wide net can degrade the user experience considerably, not to mention the time and resources wasted ranking irrelevant results. Chances are that if your top search results consist of few relevant things and many irrelevant things, you'll refine your query rather than search through pages of results.

Perhaps the occurrence of irrelevant results could be reduced if we could eliminate the unimportant content from each webpage while indexing. Thus, instead of casting such a wide net and sorting through what we caught later, maybe we can make the net smarter and more selective. Here, I investigate the feasibility of using automated document summarization, specifically Michaela et al's TextRank algorithm, and spectral clustering to do just that. Indexing only the important content may increase the precision of returned results, while also reducing index file sizes and query times. But having a smaller, more

selective net means relevant results might be missed, reducing the recall of the search engine. The extent to which these effects occur is the topic of this thesis.

Chapter 2

Background

2.1 Search Engines and Term-document Matrices

Many search engines employ a term-document matrix for conducting queries. Terms are simply the words from a document with the suffixes removed so that words like management, manager, managed all become manag. The word document is used generically and can refer to any text like a webpage, a book, or even a blog post. In this matrix, the terms from the document comprise the rows, the documents comprise the columns, and each matrix entry is the number of occurrences of the term in the document. Results are retrieved by computing the distance from the query vector, regarded as a pseudo-document, to each document column vector in the index. Depending on the Boolean operators implicitly or explicitly defined between query terms, documents that contain any or all of the query terms are returned. Results are then ranked by some measure of relevance. Relying on only word frequencies, these term-document matrices perform surprisingly well at retrieving relevant documents [1]. The success of major web search companies demonstrates their utility.

While these term-document matrices already tend to perform well in retrieving relevant documents, they can be improved using simple techniques like removing frequently occurring words that have low information content like articles, pronouns, and prepositions [1]. These are referred to as stop words. But even after stop words are removed, many of the terms in the document may not be immediately relevant to the main idea or topic, and thus have low information. This could be one cause of irrelevant documents being returned.

Finding the important content in a document is precisely what people do when they summarize. Perhaps removing all content except a representative summary will increase search engine precision much as the removal of stop words does.

2.2 Automated Summarization and TextRank

Automated document summarization is a common task in Natural Language Processing with two very different possible approaches. The more difficult approach, most akin to how humans create summaries, paraphrases the most important parts of the document. This task is difficult, as it requires an understanding of the text to create a readable summary from scratch.

An alternative approach is extractive summarization. Here the most important sentences are extracted from the document and pieced together to provide a summary. While this approach may not be as desirable as creating summaries from scratch, it can be implemented much more easily because it does not require a true understanding of the

text. It only requires being able to identify characteristics of sentences that would be good candidates for the summary. This may be done by relying on algorithms and the underlying structure and statistical properties of language.

Many approaches to extractive summarization exist. Some common techniques rely on supervised algorithms that require labeled training data to model the properties that make sentences good candidates for the summary. However, training data may not exist and can be very expensive to obtain, particularly if the documents are very specialized. Unsupervised methods are more desirable because they can be applied to any new data without the need for training data or new models.

Over the last decade, unsupervised graph-based methods have been applied to the problem extractive summarization. These methods regard a document as a graph, with some unit of text representing each vertex and the edges corresponding to some measure of similarity between each vertex. A particularly interesting implementation of one such approach is Mihalcea et al's TextRank algorithm. TextRank lets the document tell us what content is most relevant. It can be applied to different units of text in a document, but here we use the sentences from each document as the vertices. This has the advantage of producing human summaries that could be incorporated into result descriptions.

TextRank measures graph centrality much like Google's PageRank algorithm. PageRank ranks web pages based on the links between them, regarding each incoming link as a "recommendation" or "vote" for that webpage [2]. Similarly, the sentences in

TextRank "recommend" each other or "vote" for each other based on the common words that occur between them (after stemming and removal of stop words). By extracting the most "recommended" sentences from the document, reasonable summaries can be created [2].

To apply the TextRank algorithm, first the document must be parsed into sentences. Once parsed, a new vertex is created for each sentence. To score the similarity among all vertices, stop words are removed from each sentence and the words are stemmed. Each sentence's similarity is defined using Mihalcea et al's metric, which counts the number of common words between the two sentences divided by the length of the two sentences. It's defined as follows, where w is the k^{th} stemmed word of all the words occurring between the two sentences:

$$Similarity(Si, Sj) = \frac{|\{w_k | w_k \in S_i \& w_k \in S_j\}|}{\log(|S_i|) + \log(|S_j|)} \text{ [2]}$$

After scoring each sentence, the sentences with the highest similarity metrics are extracted to create the summary up to some sentence or word count threshold.

Below is the abstract from Mihalcea et al's paper introducing TextRank followed by the extractive summary of the paper generated by selecting the top two scoring sentences after applying the algorithm [2].

Abstract:

In this paper, we introduce TextRank – a graph based ranking model for text processing, and show how this model can be successfully used in natural language

applications. In particular, we propose two innovative methods for keyword and sentence extraction, and show that the results obtained compare favorably with previously published results on established benchmarks.

TextRank generated summary:

In the following, we investigate and evaluate the application of TextRank to two natural language processing tasks involving ranking of text units: (1) A keyword extraction task, consisting of the selection of keyphrases representative for a given text; and (2) A sentence extraction task, consisting of the identification of the most "important" sentences in a text, which can be used to build extractive summaries. Applying a similar line of thinking to lexical or semantic graphs extracted from natural language documents, results in a graph-based ranking model that can be applied to a variety of natural language processing applications, where knowledge drawn from an entire text is used in making local ranking/selection decisions.

In applying the TextRank algorithm, there is a danger that the summary may only consist of very similar sentences. For instance, if a document contains two highly similar sentences that differ by only a word or two, their number of votes will not be very different. Indexing both sentences will not add much more information. Further, if the document consists of two somewhat different main ideas, the summary may fail to capture both. This might be corrected by separating the sentences related to each idea first and then running the TextRank algorithm on each set of sentences.

2.2 Spectral Clustering

Given that TextRank builds a graph of the sentences in the document, spectral clustering is a natural candidate for splitting the graph into sub-graphs that have low similarity between different partitions but high similarity within partitions. Spectral clustering tries to solve a problem known as normalized cuts. That is, how can the graph be split into k sub-graphs while minimizing the sum of the weights of the edges that are cut when

splitting up the graph [3]. Normalized cuts tends to preferring partitions that are roughly equal in size [3].

An additional interpretation of spectral clustering involves random walks along the transition probability matrix of a Markov process [3]. If we were to take a random walk along the graph, moving across vertices (or states) with probability proportional to the similarity between vertices, spectral clustering is akin to finding the *k* partitions where the probability of transitioning between partitions is low [3].

Over the last decade, spectral clustering has become very popular and has been used successfully on many data sets where traditional methods such as k-means performs poorly, such as situations where the clusters do not form convex regions or blobs [3]. Figure 1 shows one such situation, comparing the results of k-means and spectral clustering algorithms on concentric circles. Spectral clustering manages to group each of the individual circles. This is the way a person would likely group the data, but this could not be achieved by k-means. I chose to test this method over k-means because in high-dimensional space like the one defined by the TextRank similarity matrix, it is difficult to know whether our data would satisfy this convex region criteria.

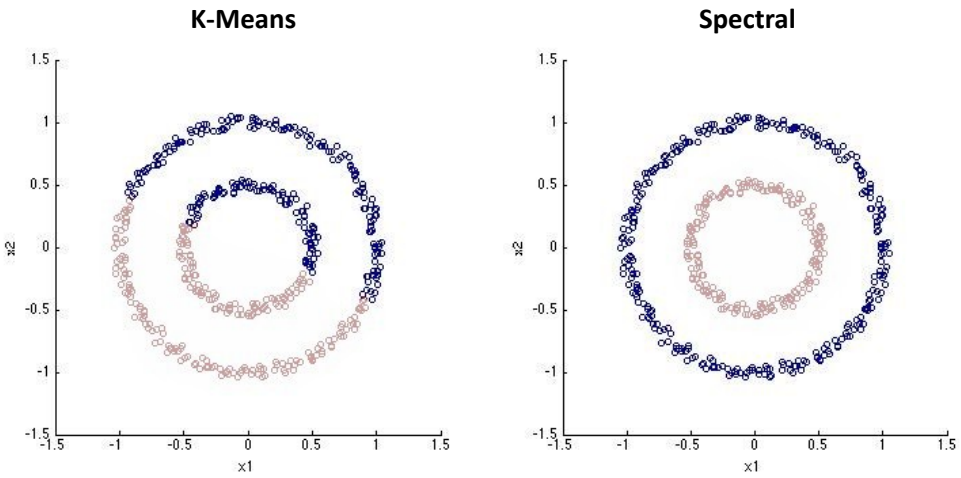

Figure 1. K-means compared to Spectral Clustering on non-convex regions with k = 2.[1]

Varying methods exists to perform spectral clustering [3]. I adopt a method based on the approach proposed by Ng et al [4]. To run their spectral clustering algorithm, an affinity matrix is constructed for every point on the graph such that the i,j^{th} element of the matrix is a measure of similarity between i^{th} and j^{th} points. Ng et al define the weight between two edges in the affinity matrix as:

$$W_{i,j} = \exp\left(\frac{-\|x_i - x_j\|^2}{2\sigma^2}\right) [4]$$

Here, σ controls the decay of the weight with increasing distance. Ng et al then construct the normalized graph Laplacian, $L = D^{-1/2}AD^{-1/2}$. The graph Laplacian captures all the properties of the graph in a single matrix. The first k eigenvectors of L are extracted, where k is the desired number of clusters. They then normalize this matrix of the first k eigenvectors. These largest eigenvectors describe the clustering properties of the similarity matrix. Running the traditional k-means algorithm on the normalized

[1] Data obtained from http://www.ima.umn.edu/~iwen/REU/REU_cluster.html

eigenvectors removes any noise. Finally, data point *i* from the original matrix is assigned to cluster *k* if and only if the i^{th} row of the normalized eigenvector matrix was assigned to cluster *k* by the k-means algorithm [4].

Note the similarity of this affinity matrix constructed for spectral clustering to the similarity matrix constructed by the TextRank algorithm. It ascribes some weight to edges of a graph where higher weights denote higher similarity. Spectral clustering can readily be applied to the TextRank algorithm to cluster the sentence graph. To do so, I simply replace this affinity matrix, which constructs a graph of the similarity between points in $\mathbf{R}^n$, with the similarity matrix between all pairs of sentences created during construction of the TextRank graph, and run the spectral clustering algorithm. This yields *k* partitions of sentences clustered based on their similarity.

Now we have k partitions of sentences. To select sentences for inclusion in the extractive summary, the average votes for each of these clusters is computed as well as the overall average votes across all sentences. Beginning with the cluster with the highest average votes, the top sentence from each cluster is selected in decreasing order of average cluster votes if and only if the sentence has higher than the overall average votes. This additional restriction on the clustering was added to avoid selecting sentences from clusters that consisted only of outlier sentences that have little relationship to the rest of the document. The selection process continues with the second sentence in each cluster until the some threshold for the proportion of words extracted is reached or there are no more sentences in each cluster with higher than average votes. Thus, the indices produced

by the TextRank plus spectral clustering method may be smaller than those produced by TextRank alone because of the above average votes restriction.

Chapter 3

Materials and Methods

3.1 Source Code and Libraries

The Apache Lucene search engine library was used to construct the search indices and execute queries. Lucene is full-featured Java-based indexing and search package that powers search in applications from many companies including Apple, Twitter, LinkedIn, and Wikipedia [5].

Additional libraries utilized include the Java Boilerplate HTML parser was used to extract text from raw HTML. The English Sentence parser from the Apache OpenNLP Natural Language Processing library was employed to split text into sentences. Eigenvalue decomposition and eigenvector extraction for the spectral clustering algorithm was done using the JAMA Linear Algebra library. Custom classes for spectral clustering and k-means were created. Finally, the TextRank algorithm was run using custom graph and node classes, while stemming and stop-word removal were done using the Apache Lucene Standard Analyzer.

3.2 Precision and Recall

Search engines are evaluated by their precision and recall. Figure 2 illustrates the components of these measures. Precision is the proportion of the results you got back from your query that you should have gotten back, or the intersection of the two circles divided by the size of the left circle. Recall is the proportion of the results you should have gotten back from your query that you should got, or the intersection of the two circles divided by the right circle. These measures are related to Type-I and Type-II errors in statistics. The Type-I error rate of the query, or false-positive rate is equal to one minus the precision. Similarly, one minus the recall is the Type-II error rate of the query, or the false-negative rate.

Figure 2. Query Result Illustration

3.3 ClueWeb09 and TREC Query Data

Evaluating text retrieval methods can be difficult because to calculate precision and recall you must have a corpus of documents to index and a predefined set of queries, each with corresponding relevance judgments for every document. It is not sufficient to have a few hundred articles from a magazine or journal. You must also have several queries and each article must be labeled as relevant or not for each query. Thus, the few text retrieval document corpora that are available tend to be smaller and based on journal article abstracts or small excerpts of text that may not be representative of real world data.

An exception to this is the data utilized by the Text REtrieval Conference (TREC), which is sponsored by the National Institute of Standards and Technology. The conference utilizes the ClueWeb09 dataset comprising about 1 billion webpages crawled in early 2009 by Carnegie Mellon University. For the yearly conference competition, TREC releases 50 test queries along with relevance judgments for the larger ClueWeb09 dataset of 1 billion webpages as well as a smaller set of 50 million webpages. These experiments were conducted using the 2009 conference Web Track competition query relevance judgments for the 50 test queries from the 1 billion document dataset.

3.4 Experimental Methods

To examine the impact of implementing the TextRank algorithm with and without spectral clustering on a search engine, several search indices were constructed using all 18,666 webpages identified for each the 50 test queries released for the Web Track of the TREC 2009 conference (listed in the Appendix). The first index was constructed by

indexing the full text content of each webpage after some preprocessing (removing HTML). This index serves as the baseline for comparison for all other test indices since this is the approach typically taken in search implementations.

A number of parameters had to be chosen to implement TextRank with and without spectral clustering. These included the threshold of the maximum amount of content to extract and index as well as the maximum number of clusters for spectral clustering. Since the main question of interest here is whether indexing less content can increase precision, this analysis was conducted with various threshold levels for extraction while fixing the maximum number of clusters at three. That is, for each method a test index was built by extracting sentences until a specified threshold for the proportion of the words in each document had been crossed. Values of 25%, 50%, 75% were tested for this extraction threshold.

For each extraction threshold, a test search index was built using the TextRank algorithm with no clustering, extracting and indexing sentences until the threshold was crossed. An additional test index was built using both the TextRank algorithm as well as spectral clustering. This method, as noted above, extracted the top sentences from each cluster, beginning with the cluster with the highest average score and continuing to select the top sentence from each cluster with decreasing average score if the sentence had higher than the overage average number of votes. This continued with the sentences with the second highest number of votes in each cluster and so on until the threshold was crossed or there were no more sentences with higher than average votes. Finally, another index was constructed by randomly sampling sentences from each document to until the

threshold was crossed. While the Full Content index serves as the baseline, this index serves as a control. Comparing results from the test indices with the randomly selected sentences index will help determine whether any changes in precision, recall, and query time over the Full Content index could be achieved by reducing the amount of content indexed arbitrarily. This ensures that any changes are correctly attributed to the method that caused them.

With the Full Content index baseline, three Random Sentences indices as the controls, one for each of the extraction thresholds, and three indices for each of the extraction thresholds for the TextRank test method and the TextRank plus spectral clustering method, there are ten indices in total. To test the impact of each approach, all 50 queries were run on each index and the query time, precision, and recall of each query were recorded. The precision and recall were calculated using the relevance judgments from the TREC 2009 web track. Additionally, the sizes of the indices and average query search times were recorded. All test of statistical significance across queries were performed using paired t-tests assuming unequal variances.

Chapter 4

Results

Figure 3 illustrates each index size in megabytes by the extraction threshold and indexing method.

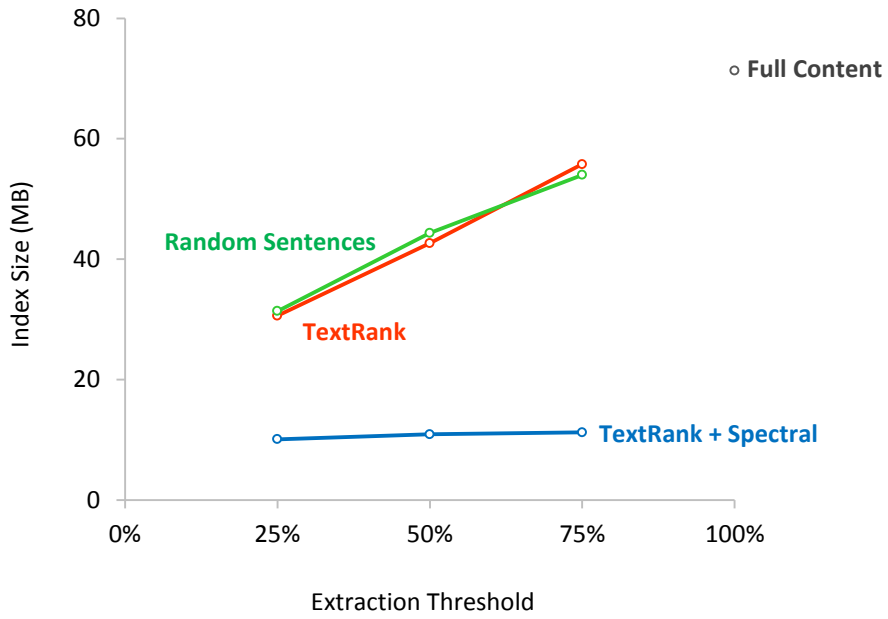

Figure 3. Index Size (MB) by Extraction Threshold and Indexing Method

The Full Content index (which corresponds to an extraction threshold of 100%) was the largest at 71.3 MB. The TextRank and Random Sentences indices were nearly identical and varied significantly with the extraction threshold at approximately 11 MB for the 25% threshold, 43 MB for the 50% threshold, and 56 MB for the 75% threshold. The

TextRank plus spectral clustering index was much smaller across all extraction thresholds ranging 10 MB for the 25% threshold to only 11 MB for the 75% threshold. This indicates that the additional restriction that for a sentence to be extracted from a cluster it must have higher than average votes is the binding constraint on index size for TextRank plus spectral clustering, not the extraction threshold. Thus, the TextRank plus spectral clustering method index size is independent of the extraction threshold.

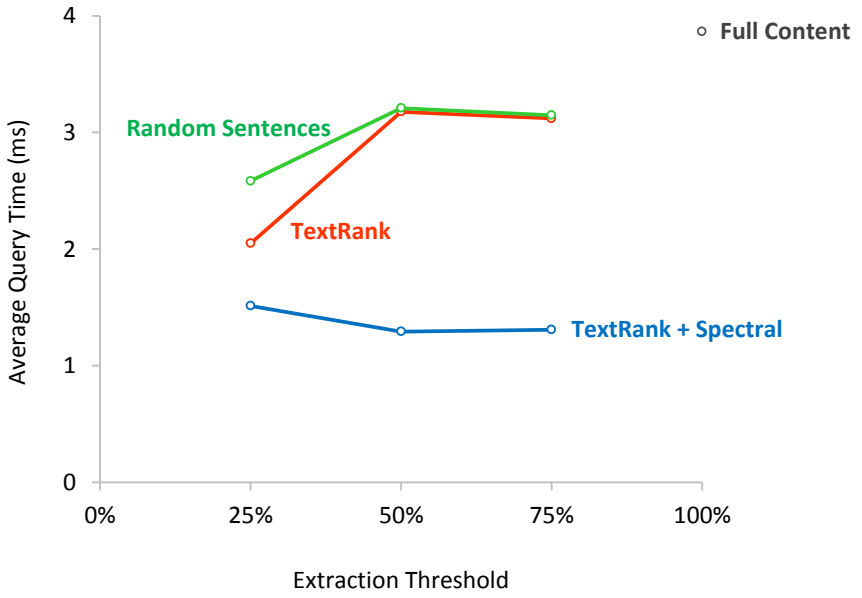

Figure 4. Average Query Time (ms) by Extraction Threshold and Indexing Method

Average query time greatly differed across methods and extraction thresholds. On average, queries on the Full Content index took 3.86 milliseconds (ms) (95% C.I.: ±0.53). There was no significant difference between average query times for the TextRank and Random Sentences indices with extraction proportions of 50% and 75%. However, the TextRank index had significantly lower average query time for the 25% threshold at 2.05 ms (95% C.I.: ±0.46) compared to 2.58 ms (95% C.I.: ±0.63). The TextRank plus spectral

clustering indices had significantly lower average query time compared to all other indices across all extraction thresholds averaging 1.37 ms (95% C.I.: ±0.08). There were no significant differences in average query times among TextRank plus spectral clustering indices.

The Full Content index had average precision of 23.8% (95% C.I.: ±0.11%) (Figure 5). That is, one in every four results was relevant. The extraction proportion had no significant impact on the precision of each indexing method. Thus, all indices were compared by averaging over all extraction thresholds.

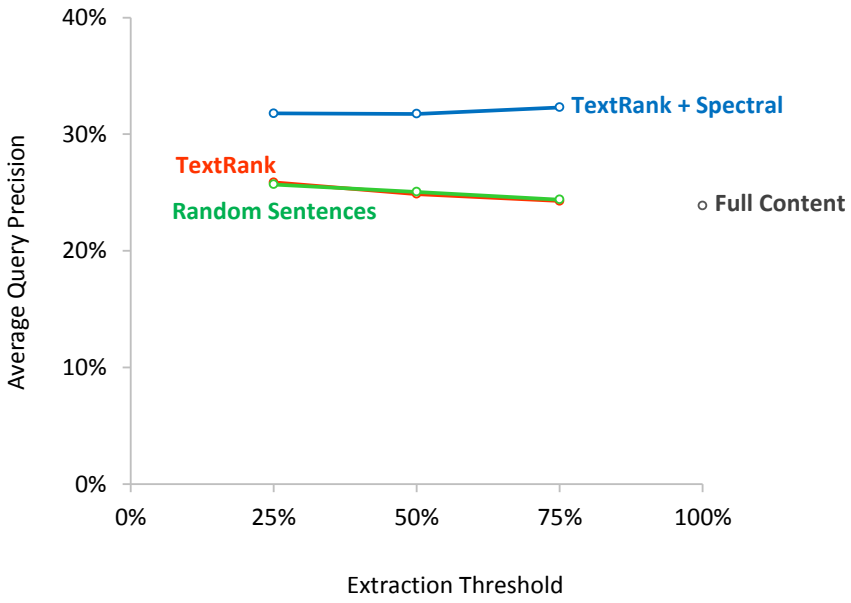

Figure 5. Average Query Precision by Extraction Threshold and Indexing Method

There were no statistically significant differences in average query precision among the TextRank, Random Sentences, or Full Content indices. However, the TextRank plus spectral clustering index had significantly higher precision at 31.9% (95% C.I.: ±0.6%)

on average compared to the TextRank index at 25.0% (95% C.I.: ±0.6%) and the Random Sentence index at 25.0% (95% C.I.: ±0.6%). This is a lift in precision of 6.9% (95% C.I.: ±3.8%) over both indices. The TextRank plus spectral clustering index also had 8.1% (95% C.I.: ±5.4%) significantly higher precision than the Full Content index.

If the relevancy scoring ranked relevant results higher, the average precision of the top ten results should be at least as high as the precision of the top twenty, and so on. In the worst case, the relevant results would be distributed more heavily among the middle or lower ranked results. Unfortunately, this is the case for all indices (Figure 6). Precision decreases as fewer top ranked results are retrieved. The TextRank plus spectral clustering method does not correct this. However, the lift in precision for the TextRank plus spectral clustering method is constant across the top ten, top twenty, and top fifty results.

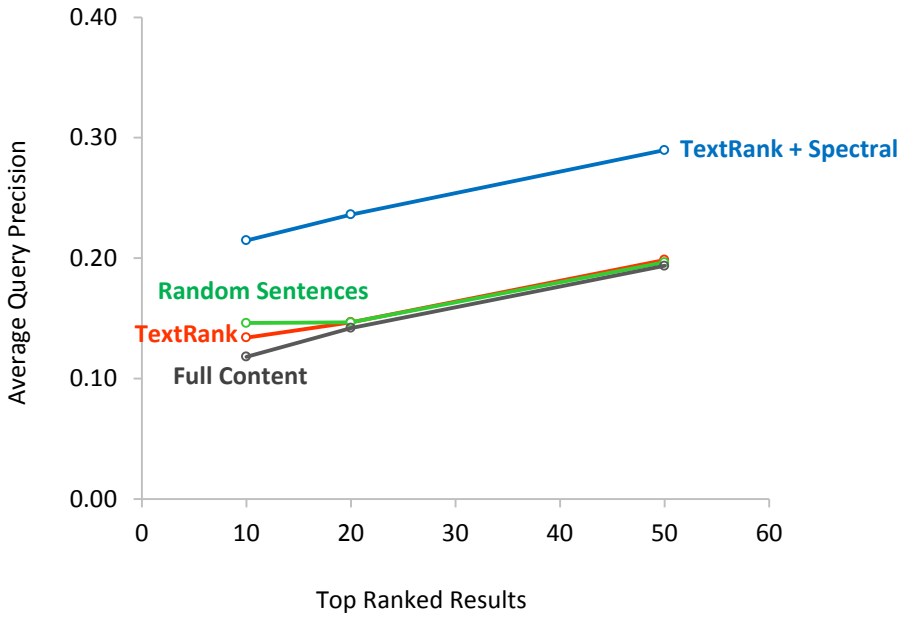

Figure 6. Average Query Precision by Top Ranked Results and Indexing Method

The Full Content index had average query recall of 89.3% (95% C.I.: ±2.2%) (Figure 7). The extraction proportion had a significant positive effect on recall for all test indices. The TextRank and Random Sentences saw similar recall for the 50% and 75% extraction thresholds at approximately 83% and 86% respectively. However, the TextRank method saw significantly higher recall compared to the Random Sentences index at an extraction threshold of 25% at 77.2% (95% C.I.: ±1.1%) compared to 71.3% (95% C.I.: ±1.1%).

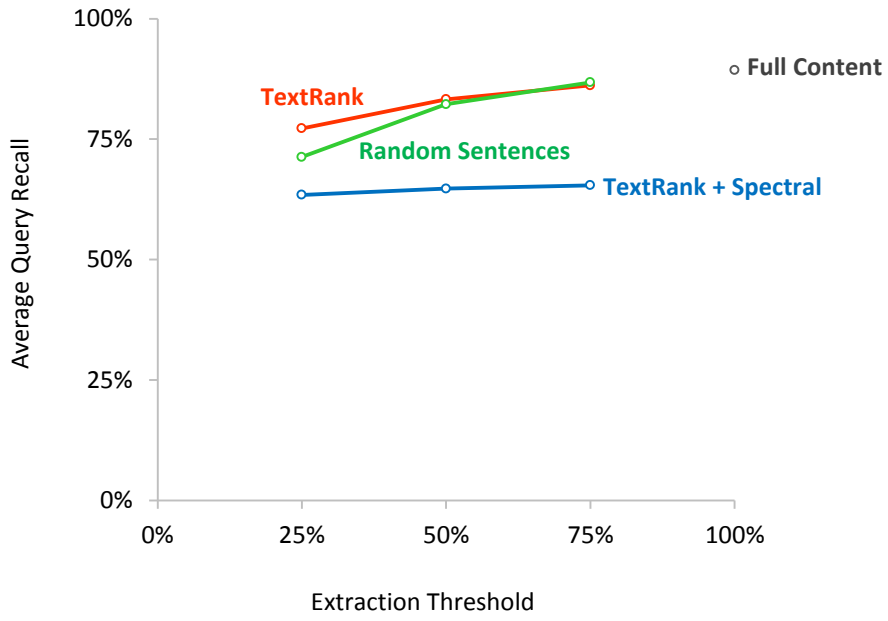

Figure 7. Average Query Recall by Extraction Threshold and Indexing Method

The TextRank plus spectral clustering indices saw significantly lower average recall compared to all other indices across all values of the extraction threshold, ranging from 63.4% (95% C.I.: ±1.1%) at an extraction threshold of 25% to 65.4% (95% C.I.: ±1.1%) at an extraction threshold of 75%. The increases in the extraction threshold tended to lead to significant increases in recall for the TextRank plus spectral clustering index, but the

effect was much smaller compared to the effect of the extraction threshold on the recall of the TextRank and Random Sentences indices.

Given that the TextRank plus spectral clustering index had significantly lower recall, were the relevant results that the Full Content index recalled but TextRank plus spectral clustering index failed to recall more or less relevant? To determine this, the relevancy score computed by the Apache search library for each search result was used. This score ranks results based on the frequency of each query term in each result multiplied by the inverse of the term's frequency of occurrence across all documents, also normalizing for the number of terms across all documents (see Appendix A for computation details). Thus, documents where the query terms occur more frequently relative to all documents that have the term score higher.

Since the relevancy score is only meaningful relative to the set of documents returned by the same query, not across indices, the scores resulting from indexing the Full content were used for comparison across all indices. Since all indexing methods use less content than the Full Content index, they could only return a subset of the results returned by the Full Content index. These scores were averaged for relevant results that were recalled by each test indexing method and recalled by the Full Content index. These scores were then compared to the average relevance scores for relevant results that failed to be recalled by each test indexing method but were recalled by the Full Content index. Paired t-tests were used to test the statistical significance of differences in the average relevancy score for each index.

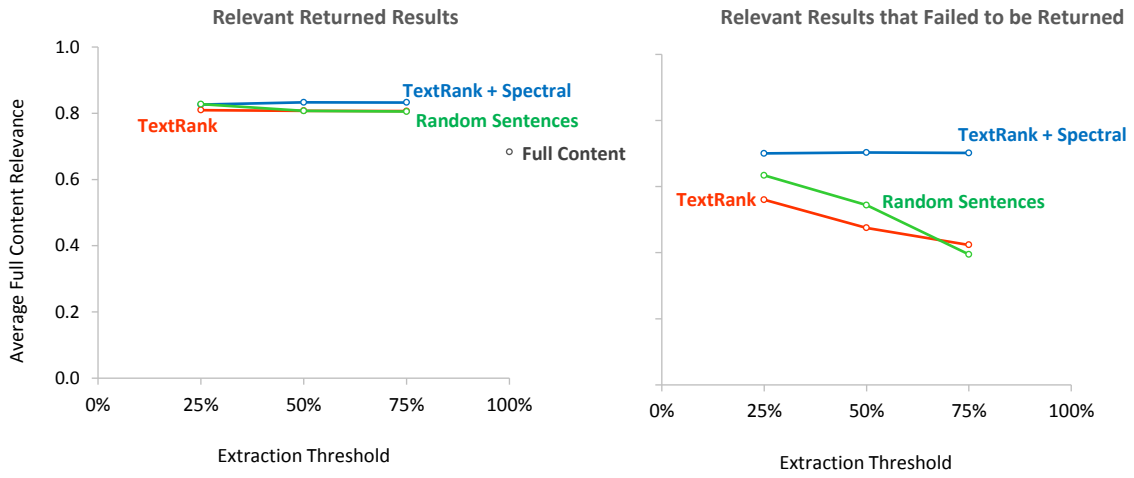

Figure 8. Comparison of Average Full Content Index Relevance for Relevant Returned Results and Relevant Results that Failed to be Returned

The average relevance scores of results that the TextRank plus spectral clustering indices recalled were significantly higher than the corresponding average relevance score of results that were not recalled by 0.13 (95% C.I.: ±0.04) (Figure 8). Thus, results that were lost due to the lower recall tended to be less relevant. This was also true of the TextRank and Random Sentences indices, where the number of results not recalled was much lower.

Also of interest is the average number of results returned relative to the average number of relevant results (Figure 9). The Full Content index returned 529 (95% C.I.: ±74) results on average. However, on average across all queries, there were only about 114 (95% C.I.: ±18) relevant results on average. The TextRank and Random Sentences indices returned similar numbers of results ranging from approximately 380 (95% C.I.: ±66) at the 25% extraction threshold to approximately 500 (95% C.I.: ±76) at the 75%

extraction threshold. TextRank plus spectral clustering returned on average about 275 (95% C.I.: ±51) results per query.

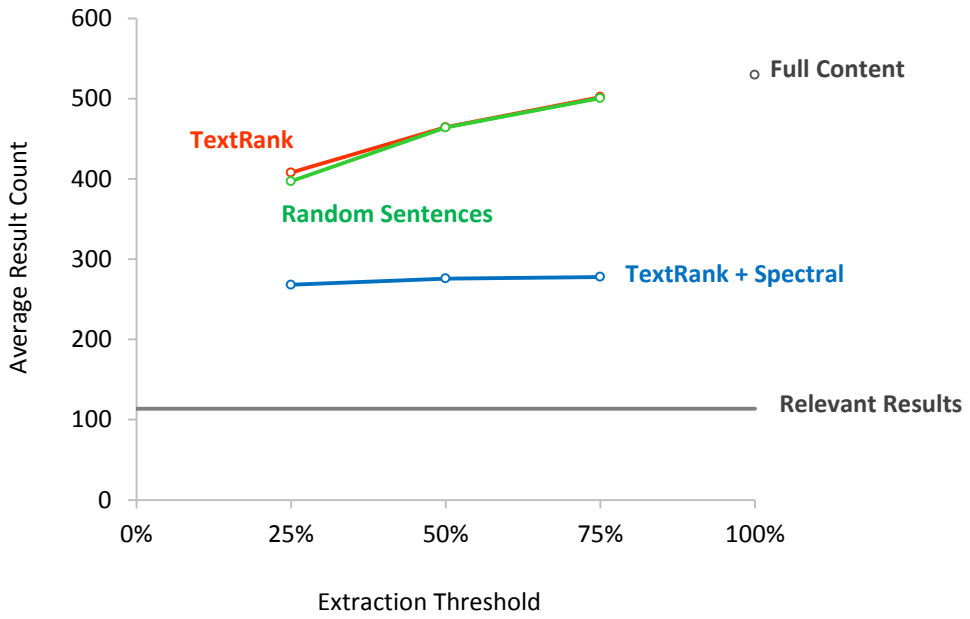

Figure 9. Average Result Count

Chapter 5

Discussion and Future Work

The evidence indicates that indexing with TextRank alone is no more effective than indexing randomly selected sentences with regard to query precision, recall, index size, and average query time. Thus, TextRank alone is not useful for finding the most important content to index in a webpage. But when combined with spectral clustering, TextRank led to a statistically significant increase in query precision of 8.1% (95% C.I.: ±5.4%) over the Full Content index. But this gain came at a price; recall was also significantly decreased by -24.8% (95% C.I.: ±7.9%) on average. However, considering that the drop in average query recall was driven by lower relevance results along with the massive decrease in index size of -86% and decrease in average query time of -64.5% (95% C.I.: ±12.1%), these losses in recall are mitigated.

The extraction threshold had no significant effect of the precision of the TextRank plus spectral clustering indices. This combined with the fact a higher extraction threshold was associated with a significant increase in average recall and the fact that it was not the binding constraint on what was indexed indicates that it can be eliminated from the method's algorithm. In fact, without the extraction threshold the method may perform better. Thus, future work could focus on testing methods other than indexing only

sentences with above average votes. Perhaps a percentile or break in the distribution of votes is more appropriate.

The maximum number of clusters for spectral clustering was fixed to three in this investigation. Less naïve methods exist for actively learning the appropriate number of clusters [5], which could lead to further improvements in the algorithm. Also, as noted by the creators of TextRank, the quality of results for the TextRank algorithm applied to sentences is heavily dependent on the quality of the sentence parsing of the documents [2]. Here, the Apache OpenNLP default English sentence parse was used. Further investigation could experiment with different parsers or perhaps sentence models customized for the web.

The methods investigated in this paper focus on reducing the amount of content indexed. TextRank with spectral clustering methods could be applied to boosting particular sentences for computing relevance scores instead. This would prevent any drops in recall. However, the additional benefits realized here such as smaller indices and faster queries would not be realized.

Chapter 6

Conclusion

TextRank plus spectral clustering increases the precision of search results. It is also more efficient with smaller index sizes compared to indexing all the content on a webpage. In exchange for possibly missing some lower relevance results, users get fewer, more precise results faster. TextRank plus spectral clustering transfers computational effort from query time, where it is spent on ranking many irrelevant results, to indexing time where it is used to find the most important content. This makes the search engine more responsive and the user interaction more fluid.

Attempts to index only important content may not be appropriate for every domain, particularly where there is a low level of information redundancy. This is surely not true of the web where content is posted, reposted, linked, and recycled and where a single webpage can contain content on many topics. When users ask you to find the needle in the haystack, they expect you to find the needle, not a smaller haystack. Applying TextRank plus spectral clustering to search is a step toward achieving just that.

Bibliography

[1] Turney, P., & Pantel, P. From Frequency to Meaning: Vector Space Models of Semantics. *Journal of Artificial Intelligence Research*, 37 pp. 141-188.2010.

[2] Mihalcea, R., & Tarau, P. TextRank – bringing order into texts. *In Proceedings of the Conference on Empirical Methods in Natural Language Processing (EMNLP 2004)*, Barcelona, Spain, 2004.

[3] Luxburg, U. A Tutorial on Spectral Clustering. *Statistics and Computing* 17 (4). 2007

[4] Ng, A.Y., Jordan, M.I., & Weiss, Y. On Spectral Clustering: Analysis and an algorithm. *NIPS* 14, 2002.

[5] The Apache Software Foundation. *Apache Lucene*. Web. 12 Mar. 2012. <http://lucene.apache.org/>.

[6] Hamerly, G., & Elkan C. Learning the K in K-Means. *NIPS* 17, 2003.

[7] Hatcher, E. & Gospodnetic, O. *Lucene in Action*. Manning Publications Co. 2005.

Appendix A

Apache Lucene Relevance Score

$$\sum_{t \in q} tf(t \text{ in } d) \cdot idf(t) \cdot boost(t.field \text{ in } d) \cdot lengthNorm(t.field \text{ in } d)$$ [6]

where

$tf(t \text{ in } d)$: Term frequency

$idf(t)$: Inverse document frequency of the term

$boost(t.field \text{ in } d)$: Boost of the field in each document (1.0 by default).

$lengthNorm(t.field \text{ in } d)$: Normalization value of the field that accounts for the number of terms it contains.

Appendix B

TREC WebTrack 09 Test Queries

obama family tree
french lick resort and casino
getting organized
toilet
mitchell college
kcs
air travel information
appraisals
used car parts
cheap internet
gmat prep classes
djs
map
dinosaurs
espn sports
arizona game and fish
poker tournaments
wedding budget calculator
the current
defender
volvo
rick warren
yahoo
diversity
euclid
lower heart rate
starbucks
inuyasha
ps 2 games
diabetes education
atari
website design hosting
elliptical trainer
cell phones
hoboken
gps
pampered chef
dogs for adoption
disneyland hotel
michworks
orange county convention center
the music man
the secret garden
map of the united states
solar panels
alexian brothers hospital
indexed annuity
wilson antenna
flame designs
dog heat

Appendix C

Supplemental Tables

Table 1. Average Query Time (ms) by Index

Method						Average Query Time	Std. Error
Full	A					3.86	0.06
Random50		B				3.21	0.03
TextRank50		B				3.18	0.06
Random75		B				3.15	0.19
TextRank75		B				3.12	0.17
Random25			C			2.58	0.14
TextRank25				D		2.05	0.13
TextRankSpectral25					E	1.51	0.23
TextRankSpectral75					E	1.31	0.14
TextRankSpectral50					E	1.29	0.13

Levels not connected by same letter are significantly different, α = 0.05.

Table 2. Mean Precision by Index

Method					Mean Precision	Std. Error
TextRankSpectral75	A				32.3%	0.5%
TextRankSpectral25	A				31.8%	0.5%
TextRankSpectral50	A				31.7%	0.5%
TextRank25		B			25.9%	0.5%
Random25		B	C		25.7%	0.5%
Random50		B	C	D	25.1%	0.5%
TextRank50		B	C	D	24.9%	0.5%
Random75		B	C	D	24.4%	0.5%
TextRank75			C	D	24.3%	0.5%
FullContent				D	23.9%	0.5%

Levels not connected by same letter are significantly different, α = 0.05.

Table 3. Mean Precision by Index and Top 10, Top 20, and Top 50 Results

Method					Mean Precision	Std. Error
TextRankSpectral, Top 50	A				0.29	0.02
TextRankSpectral, Top 20		B			0.24	0.02
TextRankSpectral, Top 10		B			0.21	0.02
TextRank, Top 50		B	C		0.20	0.02
Random, Top 50		B	C		0.20	0.02
FullContent, Top 50		B	C	D	0.19	0.03
TextRank, Top 20				D	0.15	0.02
Random, Top 20				D	0.15	0.02
Random, Top 10				D	0.14	0.02
FullContent, Top 20			C	D	0.14	0.03
TextRank, Top 10				D	0.13	0.02
FullContent, Top 10				D	0.12	0.03

Levels not connected by same letter are significantly different, α = 0.05.

Table 4. Mean Recall by Index

Method							Mean Recall	Std. Error
FullContent	A						89.3%	1.1%
Random75	A						86.8%	1.1%
TextRank75	A	B					86.1%	1.1%
TextRank50		B	C				83.3%	1.1%
Random50			C				82.2%	1.1%
TextRank25				D			77.2%	1.1%
Random25					E		71.3%	1.1%
TextRankSpectral75						F	65.4%	1.1%
TextRankSpectral50						F	64.7%	1.1%
TextRankSpectral25						F	63.4%	1.1%

Levels not connected by same letter are significantly different, α = 0.05.

Table 5.Mean Relevance of Recalled Results

Method					Mean Relevance	Std. Error
TextRankSpectral50	A				0.83	0.01
TextRankSpectral75	A				0.83	0.01
Random25	A	B			0.83	0.01
TextRankSpectral25	A	B	C		0.83	0.01
FullContent	A	B	C	D	0.82	0.01
TextRank25		B	C	D	0.81	0.01
TextRank50			C	D	0.81	0.01
Random50			C	D	0.81	0.01
TextRank75				D	0.81	0.01
Random75				D	0.81	0.01

Levels not connected by same letter are significantly different, α = 0.05.

Table 6. Mean Relevance of Results that Failed to be Recalled

Method						Mean Relevance	Std. Error
TextRankSpectral50	A					0.70	0.03
TextRankSpectral75	A					0.70	0.03
TextRankSpectral25	A					0.70	0.03
Random25	A	B				0.63	0.03
TextRank25		B	C			0.56	0.03
Random50			C			0.54	0.03
TextRank50			C	D		0.47	0.03
TextRank75				D		0.42	0.03
Random75				D		0.39	0.03
FullContent					E	0.00	0.03

Levels not connected by same letter are significantly different, α = 0.05.

Table 7. Mean Relevance Difference between Recalled and not Recalled Results

Method					Mean Relevance Difference	Std. Error
Random75	A				0.41	0.03
TextRank75	A				0.38	0.03
TextRank50	A	B			0.34	0.03
Random50		B	C		0.26	0.03
TextRank25			C		0.25	0.03
Random25			C	D	0.19	0.03
TextRankSpectral75				D	0.13	0.03
TextRankSpectral50				D	0.13	0.03
TextRankSpectral25				D	0.13	0.03

Levels not connected by same letter are significantly different, α = 0.05.

Table 8. Mean Results by Index

Method					Mean Results	Std. Error
FullContent	A				529.78	37.92
TextRank75	A				501.98	38.87
Random75	A				500.62	38.69
TextRank50	A	B			464.36	36.48
Random50	A	B			463.88	37.39
TextRank25		B			407.88	33.22
Random25		B			396.96	33.91
TextRankSpectral75			C		277.76	26.20
TextRankSpectral50			C		275.74	26.06
TextRankSpectral25			C		268.10	26.21
Relevance Results				D	113.68	9.08

Levels not connected by same letter are significantly different, α = 0.05.